(5)

(6) # THE NAVY ~~FORTRAN~~ VALIDATION SYSTEM

(10) by Patrick M./Hoyt
Department of the Navy
Washington, D. C.

(11) 9 May 77

(12) 27p.

(14) FCCTS/TR-77/18

ABSTRACT

The FORTRAN Compiler Validation System (FCVS) developed by the Department of the Navy tests the conformance of those elements of the FORTRAN language which are contained in the logical intersection of the American Standard FORTRAN, X3.9-1966, and the elements proposed for the subset language in the draft proposed American National Standard Programming Language FORTRAN.

This paper discusses the development of the FORTRAN Compiler Validation System and presents the rationale for the FCVS. The design criteria for the FCVS and a description of the test production is explained. The capabilities of the Executive System are described as well as the future developments anticipated for the FCVS because of the adoption of the revised FORTRAN Standard and the impact of the CODASYL FORTRAN Data Base Facility.

Federal Cobol Compiler Testing Service D.C.

408 438 N.

BIBLIOGRAPHIC DATA SHEET	1. Report No. FCCTS/TR-77/18	2.	3. Recipient's Accession No.
4. Title and Subtitle The Navy FORTRAN Validation System			5. Report Date 9 May 1977
			6.
7. Author(s) Patrick M. Hoyt			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Software Development Division Department of the Navy ADPE Selection Office Washington, D. C. 20376			10. Project/Task/Work Unit No.
			11. Contract/Grant No.
12. Sponsoring Organization Name and Address ADPE Selection Office Department of the Navy Washington, D. C. 20376			13. Type of Report & Period Covered
			14.
15. Supplementary Notes			

16. Abstracts

The FORTRAN Compiler Validation System (FCVS) developed by the Department of the Navy tests the conformance of those elements of the FORTRAN language which are contained in the logical intersection of the American Standard FORTRAN, X3.9-1966, and the elements proposed for the subset language in the draft proposed American National Standard Programming Language FORTRAN.

This paper discusses the development of the FORTRAN Compiler Validation System and presentssthe rationale for the FCVS. The design criteria for the FCVS and a description of the test production is explained. The capabilities of the Executive System are described as well as the future developments anticipated for the FCVS because of the adoption of the revised FORTRAN Standard and the impact of the CODASYL FORTRAN Data Base Facility.

17. Key Words and Document Analysis. 17a. Descriptors

FORTRAN
Validation
Software
Audit Routines
Verifying
Compilers
Standards
Programming Languages

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement Release Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 26
	20. Security Class (This Page UNCLASSIFIED	22. Price

INTRODUCTION

FORTRAN is one of the oldest of the higher level programming languages with its roots in IBM in 1954[1]. Standardization for the FORTRAN language began in May 1962 under the direction of the American Standards Association Committee X3.4.3.* In 1966, two standards were published for the FORTRAN language: American Standard FORTRAN, X3.9-1966[2] and American Standard Basic FORTRAN, X3.10-1966, which is a proper subset of the first Standard.

---

*The American Standards Association (ASA) has since changed its name to the American National Standards Institute, Inc. (ANSI). The FORTRAN Committee is now known as X3J3.

The FORTRAN Compiler Validation System (FCVS) developed by the Department of the Navy tests the conformance of those elements of the FORTRAN language which are contained in the logical intersection of the American Standard FORTRAN, X3.9-1966, and the elements proposed for the subset language in the draft proposed American National Standard Programming Language FORTRAN[3].

One of the principal reasons for developing validation systems is the principal criteria given for developing the FORTRAN Standard: "Interchangeability of FORTRAN programs between processors"[3]. The FCVS was developed as a tool to enable users to

acquire FORTRAN compilers which meet the ANSI language specifications. The availability of FORTRAN compilers conforming to the Standard enhances the interchangeability of FORTRAN programs.

The FCVS consists of FORTRAN audit routines, their related test data, and an executive routine (EXECUTIVE) which prepares the audit routines for compilation and execution. Each audit routine consists of series of tests of FORTRAN language elements, and supporting procedures which indicate the result of executing these tests. Because the routines were designed to run on any computer system purporting to support FORTRAN, the assumptions used to write the audit tests are very restrictive. Only the simplest forms of GO TO, Arithmetic IF, WRITE, and assignment statements are used to write the support code required for each test. A complete discussion of the FCVS test philosophy and a full description of each of the language element tests are contained in the document FCVS DETAILED TEST SPECIFICATIONS.[4]

A SOURCE PROGRAMS file of audit routines with appropriate implementor-defined parameters inserted into the source code is produced by the EXECUTIVE. The EXECUTIVE is a FORTRAN program included in source form in the FCVS LIBRARY. Once installed, the EXECUTIVE is used each time that an audit routine or series of audit routines is selected from the FCVS LIBRARY. Basic inputs to this process are the FCVS LIBRARY (a file of all of the audit routines, the EXECUTIVE and related test data), and a series of control inputs to select and/or update the audit routine source code.

A FORTRAN compiler, in a particular computer configuration/operating system environment, is tested by the compilation and execution of each audit routine. If a compiler rejects some language element by giving fatal diagnostic messages or terminating the compilation, then the EXECUTIVE is used to eliminate the source code containing that language element. The audit routine is then recompiled and executed. Output reports (TEST RESULTS) produced by the execution of each routine indicate whether the code generated by the compiler passed or failed each test of the routine. The TEST RESULTS together with the compilation listings constitute the raw data from which the Department of the Navy produces a Validation Summary Report (VSR). The VSR itemizes the areas where the FORTRAN compiler being tested does not conform with the American National Standard FORTRAN specifications.

HISTORY

A study of available FORTRAN validation systems was performed in August 1973. This study analyzed the U. S. Navy FORTRAN tests developed by Captain Grace Hopper of the Navy Programming Languages Section[5], and the National Bureau of Standards FORTRAN tests developed by F. E. Holberton and E. G. Parker[6]. The study concluded that the major flaws in these validation routines were that all the test results were listed on a printer and required careful examination of the test results by the user, and these test routines required many manual changes to the source code when preparing them for execution on a given computer system.

At this time it was decided that the FCVS developed by the Software Development Division must evaluate the results of the language tests within the tests themselves, and print PASS or FAIL for each test in the same manner as the COBOL Validation System. In 1973 a three stage project was designed to:

(1) extract and modify existing tests and routines;

(2) add PASS/FAIL/DELETE support code to make the routines self-measuring; and

(3) build a complete FORTRAN validation system based on a set of simple assumptions and the self-measuring techniques used in implementing the second stage.

Due to lack of available resources, the FCVS project remained in abeyance until February 1975 when the decision was made to pursue the third stage as the initial effort. The scope of the FCVS project was to adequately test all of the elements of the FORTRAN language based on the specifications in American Standard FORTRAN, X3.9-1966.

DEVELOPMENT OF THE FCVS

The FCVS project was broken into five major phases as follows:

1. Systems Analysis and Design Phase -
   - develop the matrix of language elements to be tested
   - develop the list of basic test assumptions, programming and naming conventions, EXECUTIVE routine functions and requirements, test and implementation procedures.
2. Program Analysis and Design Phase -
   - produce detailed specifications for each audit routine.
3. Coding and Debugging Phase -
   - write boiler plate for TEST RESULTS format
   - code three programs to test the basic assumptions
   - code and debug an estimated thirty elementary routines
   - code and debug an estimated twenty advanced routines
   - test data to be prepared as required.
4. Integration and Testing Phase -
   - write detailed specifications for the EXECUTIVE, then code and debug the EXECUTIVE routine
   - integrate the EXECUTIVE, audit routines and any test data onto the FCVS LIBRARY
   - test the final integrated FCVS as a system.

5. Documentation and Release Phase -

   ° update all documentation to reflect final FCVS specifications then release and distribute through NTIS.

Based on this scope of the FCVS project, eighteen (18) manmonths were estimated for completion of the project. Two computer specialists were assigned to share equally the responsibilities of the entire project. It was estimated, based on the experience gained in developing the CCVS74 audit routines, that the two computer specialists could devote half of their available time to the project. The FCVS project was to begin October 1975 and was scheduled for completion on 1 July 1976.

Work proceeded on schedule until January 1976. Very little progress was made on the FCVS during January and February 1976 as the available manpower was devoted to higher priority projects. In March 1976, two major decisions were made. The number of tests in any one routine were limited to thirty (30), since the TEST RESULTS report could then be printed on a single page (approximately 56 lines). The draft proposed American Standard FORTRAN (X3J3 - pending), which had been distributed for public comment, was analyzed with respect to the language elements identified in American Standard FORTRAN, X3.9-1966. It was decided that the FCVS version 1.0 then being developed would test the conformance of those elements of the FORTRAN language which are contained in the logical intersection of American Standard FORTRAN, X3.9-1966, and the elements proposed in the subset language of the draft proposed American Standard Programming Language FORTRAN. The previous arbitrary classification of elementary versus advanced language elements was deemed obsolete since the proposed Standard contained a subset language.

The FCVS was designed to build the statement tests from a basic set of FORTRAN language features which are assumed to function correctly. The remaining language features are tested using these basic language elements. The assumptions were made with the goal that these routines would be executable on most minicomputer systems as well as on the larger computer configurations.

The basic assumptions are listed below and the references to X3.9-1966 are enclosed in parentheses.

(1) Six character symbolic names (3.5 and 10.1) and five digit statement labels (3.4) are permitted.

(2) Comment lines (3.2.1) do not affect a program in any way.

(3) Execution of the unconditional GO TO statement (7.1.2.1.1) GO TO k causes the statement identified by the statement label k to be the next statement executed.

(4) Branching to a CONTINUE statement (7.1.2.6) causes the statement following the CONTINUE statement to be the next statement executed.

(5) The assignment statements (7.1.1.1)

    integer variable = integer constant (5.1.1.1)
    integer variable = integer variable
    real variable = real constant (5.1.1.2)
    real variable = real variable

function correctly.

(6) The arithmetic IF statement (7.1.2.2) functions correctly: IF (e) k1, k2, k3 where e is an arithmetic expression (6.1) of the form

    integer variabel + integer constant
    integer variable - integer constant
    real variable + real constant
    real variable - real constant

and k1, k2, and k3 are statement labels.

(7) The simple formatted WRITE statement (7.1.3.2.3) functions correctly: WRITE (u,f) k where u is a logical unit number (7.1.3.1), f is a FORMAT statement label, and k is a list (7.1.3.2.1) of integer and real variables.

The format statement contains nH Hollerith field descriptors (7.2.3.8), nX blank field descriptors (7.2.3.9), Iw numeric field descriptors (7.2.3.6.1), and Fw.d numeric field descriptors (7.2.3.6.2).

(8) In order for the output report to have the correct format, the use of the first character of a formatted record for vertical spacing must function correctly (7.1.2.4).

Two characters which are used in printing the report are:

CHARACTER	VERTICAL SPACING BEFORE PRINTING
1	Advance to first line of next page
blank	One line

In addition to the preceding basic assumptions, the following minimum capabilities are assumed for the routines:

(1) Integer variables consist of at least 16 bits of which one is a sign bit.

(2) The system output device has at least 56 characters per line.

(3) Real variables contain at least 16 bits in the mantissa and 8 bits in the exponent.

In order to appreciate the changes in scope made during the FCVS project, it is essential that one understands what was considered elementary versus an advanced audit routine in the original identification of the tasks. The following list shows the language element areas originally chosen for elementary vs. advanced.

Also shown is a column for whether a given language element area was tested in version 1.0 of the FCVS.

LANGUAGE ELEMENT AREA	ORIGINAL LEVEL	VERSION 1.0
Comment lines	Elementary	Tested
Reference format blanks in variables statement labels continuation of lines FORTRAN reserved words	Elementary	Tested
Simple Subroutine call	Elementary	Tested
Subroutine calls another routine	Elementary	Tested
Intrinsic functions	Elementary	Tested
DATA statement	Elementary	Tested
BLOCK DATA subprogram	Elementary	Deferred to a later version
Blank COMMON	Elementary	Tested
Labeled COMMON	Elementary	Tested
EQUIVALENCE statement	Elementary	Tested
EQUIVALENCE and COMMON	Elementary	Tested
DO loops - simple format	Elementary	Tested
CONTINUE statement	Elementary	Tested
Arithmetic IF statement	Elementary	Tested
Logical IF	Elementary	Tested
Unconditional GO TO statement	Elementary	Tested

LANGUAGE ELEMENT AREA	ORIGINAL LEVEL	VERSION 1.0
Assigned GO TO	Elementary	Tested
Computed GO TO	Elementary	Tested
TYPE statement	Elementary	Tested
Integer arithmetic tests	Elementary	Tested
Arrays - fixed dimensions, simple constant and variable subscripts	Elementary	Tested
Repeated calls to a subroutine	Elementary	Tested
Inline arithmetic statement functions	Elementary	Tested
FUNCTION subprogram	Elementary	Tested
Multiple RETURN statements	Elementary	Tested
Logical data	Elementary	Tested
Logical expressions	Elementary	Tested
Simple sequential file I/O	Elementary	Tested
Character set	Elementary	Tested
Subroutines sharing COMMON	Advanced	Added**
Nested DO loops and extended range of a DO statement	Advanced	Added**
DO index tests	Advanced	Added**
Real arithmetic tests - adjustable accuracy	Advanced	Deferred
Double precision data	Advanced	Deferred
Complex data	Advanced	Deferred
Arrays - arithmetic expressions for subscripts	Advanced	Deferred

LANGUAGE ELEMENT AREA	ORIGINAL LEVEL	VERSION 1.0
EQUIVALENCE with COMMON and DIMENSION arrays	Advanced	Added**
EXTERNAL statement	Advanced	Deferred
REWIND ENDFILE BACKSPACE READ WRITE	Advanced	Added**
I/O with implied DO loops	Advanced	Added**
Variable logical unit numbers	Advanced	Added**
Binary READ and WRITE unformatted	Advanced	Deferred
Scaling in FORMAT statement	Advanced	Deferred
F, E, I FORMAT field descriptors	Advanced	Added**
Evaluation of expressions - many variables, arithmetic, relational and logical	Advanced	Deferred
Assignment rules for expressions with a change in data type	Advanced	Added**
Variable dimensioned arrays in subprograms	Advanced	Deferred
External procedure names as arguments in intrinsic function references	Advanced	Deferred

**Added elements were included in Version 1.0. Remaining advanced elements are not included in Version 1.0, however, these elements will be tested in future versions of the FCVS.

Additional manpower resources were added to the FCVS project in late April 1976 as the number of routines to be written had increased from fifty (50) to seventy-five (75).

Description of Statement Tests

The statement tests in the FCVS were built carefully from the foundation of the basic assumptions. There was a systematic increase in the complexity of the language features tested as succeeding FORTRAN audit routines were developed. Language features other than those in the basic assumptions were not included in a test until they had been thoroughtly tested themselves. This method provides for the cross checking of test failures and allows for the precise identification of problem areas due to nonconformance to the language specifications or other compiler errors and deficiencies.

The first several routines in the FCVS test the language elements in the basic assumptions. Their correct execution ensures that the failure of any test in the remainder of the routines is due to the improper implementation of the language feature being tested.

A description of the first few tests for the Arithmetic Assignment Statement is included to show how the tests build upon previous tests. An Arithmetic Assignment Statement is of the form:

variable name = arithmetic expression.

The simplest form for the arithmetic assignment statement is:

integer variable = integer constant.

The first audit routine which tests arithmetic assignment statements contains tests of the above form where the integer constant is unsigned, positively signed and negatively signed. The unsigned and positively signed constants increase in absolute value in succeeding tests to a maximum of 32767, and the negatively signed constants decrease in value to -32766.

The next form of the arithmetic assignment statement to be tested is:

integer variable = integer variable.

In order to test this form the statements from the previous tests setting an integer equal to a constant must be used. The source code lines for these tests are:

integer variable 1 = integer constant

integer variable 2 = integer variable 1,

where the integer constant assumes the values previously tested.

This process is continued with tests of arithmetic assignment statements of the form

integer variable = integer variable + constant,

integer variable = integer variable - constant,

integer variabel = integer variable + integer variable,

integer variable = integer variable - integer variable.

By developing tests in this manner, if a problem with a language element appears in a particular construct, the problem is easily identified in all other tests which employ the same type of construct.

__Test Support Code__

The tests in the FCVS contain support source code which checks the results of the language features tested and procedus output indicating the results of each test. The support source lines also contain statements which are executed if a test must be deleted in order to compile a program. If the compiler cannot handle a particular language feature which is being tested, that code is deleted by placing a C in column 1 of the source lines for that test. During execution, the program falls immediately into the test deletion source lines.

Section 9.2 of the 1966 FORTRAN Standard states "A program part may not contain an executable statement that can never be executed"[2]. Since the test deletion code is only executed when a test is deleted, this specification required several IF statements to be added to the support code. The IF statements refer to statement labels which being lines of source code which are not executed if the language element tested performs correctly.

An example of the source lines for two tests of the arithmetic assignment statement is given in Figure 1 to show the test construction and the support code common to each test. Figure 2 contains the same tests but test number 227 has been deleted in this example. In the execution of these tests on a given system, the Executive System will replace the X02 in the WRITE statement with the implementor-defined logical unit number for the printer.

Audit Routine Outout Report

The output report for each audit routine indicates whether the individual tests in the routine passed, failed or were deleted. A summary of the results for each routine is printed at the end of the output report. Figure 3 is an example of the output report for the audit routine FM004. This report shows that two tests in this routine failed, and the computed and expected results are given for these two tests. The comment lines within the program or the program documentation would have to be consulted to determine what language elements did not conform to the language specifications and thus caused these tests to fail.

The Executive System

The FCVS source library tape contains system independent source programs with implementor-defined aspects such as logical unit numbers yet to be resolved. The Executive System was developed to build compilable programs from the FCVS source library tape. The purpose of the Executive System is to handle the implementation problems which occur even with programs written in Standard FORTRAN.

The Elementary Executive Routine was written for execution on a minicomputer system and contains only those capabilities expected of a system with limited resources. Because of this, the Executive Routines are written in FORTRAN using only language elements and features included in the basic assumptions.

The Elementary Executive Routine permits the selection of a program from the FCVS source library tape by program identifier and the building of a compilable program file. Resolution of implementor-defined logical unit numbers and update capabilities by source line are performed as the program file is built. The update capabilities include inserting a source line, replacing a source line, deleting a source line, and changing a source line to a comment line by placing a C in column 1.

Testing

During July and August 1976, the audit routines comprising version 1.0 of the FCVS were tested on four systems:

- UNIVAC 1108 Field Data compiler under EXEC-8
- Data General NOVA 800 under RDOS version 3.0
- Digital Equipment Corporation PDP 11/70 under RSX-11M
- General Electric FORTRAN IV compiler under the MARK III timesharing system.

Milestones

The following chart shows the actual milestone completion dates to develop the FCVS.

Matrix to Identify FORTRAN Language Elements	31 OCT 75
Programming Procedures Document	21 NOV 75
FCVS Test Plan	26 MAR 76
EXECUTIVE Routine Specifications	28 MAY 76
FCVS Test Specifications - Working Papers	11 JUN 76
EXECUTIVE Routine Completed	26 JUN 76
Version 1.0 Test Routines Completed	04 JUL 76
FCVS Detailed Test Specifications Manual Version 1.0	09 JUL 76
Testing Completed - FCVS LIBRARY Tape Version 1.0 Produced	13 AUG 76
FCVS User's Guide Manual Version 1.0 Completed	13 AUG 76

SCOPE OF THE FCVS

The purpose of the FORTRAN Compiler Validation System is the testing of a compiler's conformance to the FORTRAN language specifications. The tests in the FCVS are "positive" in that only statements permitted by the Standard are included. There is no "negative" tests of incorrect statement formats which a compiler is suppose to flag as errors.

The FCVS also does not test vendor extensions to the language specifications, and does not perform an error analysis on the results of executing the Basic External Functions supplied by FORTRAN processors. The FCVS is not designated to measure the efficiency of the object code generated or the performance characteristics of a FORTRAN compiler.

FUTURE FCVS DEVELOPMENT

X3J3 has developed a draft proposed revised FORTRAN Standard consisting of a full language and a subset language to replace American Standard FORTRAN, X3.9-1966. X3J3 has also recommended withdrawal of X3.10-1966, Basic FORTRAN since a FORTRAN subset is defined in the revision to X3.9-1966. The proposed revision is in the process of being accepted by ANSI and it is anticipated in the "near" future there will be a new FORTRAN Standard.

A study of the draft revised Standard and associated appendicies reveals that programs conforming to the 1966 Standard will also conform to the revised Standard. The changes to FORTRAN from X3.9-1966 to the X3J3 revision were made "only when such changes were necessary to correct an error in the previous standard or to add to the power of the FORTRAN language in a significant manner. In addition, such changes were only considered when it was felt that the change would not affect a significant number of programs"[3].

The FCVS developed for the 1966 Standard will be the foundation for an FCVS for the complete revised Standard. Major additions to the current FCVS will be required to test the new language features in the revised Standard. The motivation and philosophies previously described for the current FCVS remain essentially intact in developing a compiler validation system for the complete revised language Standard.

The FORTRAN Data Base Committee of CODASYL is developing a data base facility to allow a FORTRAN user to manipulate data bases. The data base facility is based on both the CODASYL Data Base Facility and the revised FORTRAN Standard. A working document of the FORTRAN Data Base Committe, CODASYL FORTRAN Data Base Facility Journal of Development[7], describes a set of data manipulation language statements and data definition language statements "intended to be in the spirit of FORTRAN".

If the FORTRAN data base facility is accepted by the FORTRAN Community then data base validation routines would be developed for inclusion in the FCVS. The growth in the use of data base concepts for large and small scale computer systems makes validation techniques for host language interfaces important.

CONCLUSIONS

The FORTRAN Compiler Validation System provides a tool for measuring a compiler's conformance to the FORTRAN language specifications. Properly administered, the FCVS will promote improvements and eliminate compiler deficiencies from vendor supplied software. The FCVS will be used by the ADPE Selection Office, Department of the Navy, in the procurement process. It is an important addition to procurement procedures and the FCVS will ensure the selection of computer systems with compilers that support the FORTRAN Standard.

The FCVS is now available to the user community. Any comments or suggestions on the FCVS will be appreciated and should be addressed to:

Department of the Navy
Software Development Division
ADPE Selection Office
Washington, D. C. 20376

```
C
C     TEST 225 THROUGH 234 USE PARENTHESES TO GROUP ELEMENTS IN AN
C     ARITHMETIC EXPRESSION.
C

            . . .

 2271 CONTINUE
      IVTNUM = 227
C
C     ****  TEST  227  ****
C           INTEGER VARIABLE = (2 + INTEGER VARIABLE) + 4
C
      IF (ICZERO) 32270, 2270, 32270
 2270 CONTINUE
      IVON01 = 3
      IVCOMP = (2+IVON01) + 4
      GO TO 42270
32270 IVDELE = IVDELE + 1
      WRITE (X02,80003) IVTNUM
      IF (ICZERO) 42270, 2281, 42270
42270 IF (IVCOMP - 9) 22270,12270,22270
12270 IVPASS = IVPASS + 1
      WRITE (X02,80001) IVTNUM
      GO TO 2281
22270 IVFAIL = IVFAIL + 1
      IVCORR = 9
      WRITE (X02,80004) IVTNUM, IVCOMP ,IVCORR
 2281 CONTINUE
      IVTNUM = 228
C
C     ****  TEST 228  ****
C           INTEGER VARIABLE = 2 + (INTEGER VARIABLE + 4)
C
      IF (ICZERO) 32280, 2280, 32280
 2280 CONTINUE
      IVON01 = 3
      IVCOMP = 2 +(IVON01+4)
      GO TO 42280
32280 IVDELE = IVDELE + 1
      WRITE (X02,80003) IVTNUM
      IF (ICZERO) 42280, 2291, 42280
42280 IF (IVCOMP - 9) 22280, 12280, 22280
12280 IVPASS = IVPASS + 1
      WRITE (X02,80001) IVTNUM
      GO TO 2291
22280 IVFAIL = IVFAIL + 1
      IVCORR = 9
      WRITE (X02,80004) IVTNUM, IVCOMP ,IVCORR

   . . .
```

FIGURE 1

Example of Source Lines for Test of Arithmetic Assignment Statement

```
C
C      TEST 225 THROUGH 234 USE PARENTHESES TO GROUP ELEMENTS IN AN
C      ARITHMETIC EXPRESSION.
C

           . . .

 2271  CONTINUE
       IVTNUM = 227
C
C      ****  TEST  227  ****
C            INTEGER VARIABLE = (2 + INTEGER VARIABLE) + 4
C
       IF (ICZERO) 32270, 2270, 32270
 2270  CONTINUE
C      IVON01 = 3
C      IVCOMP = (2+IVON01) + 4
C      GO TO 42270
32270  IVDELE = IVDELE + 1
       WRITE (X02,80003) IVTNUM
       IF (ICZERO) 42270, 2281, 42270
42270  IF (IVCOMP - 9) 22270,12270,22270
12270  IVPASS = IVPASS + 1
       WRITE (X02,80001) IVTNUM
       GO TO 2281
22270  IVFAIL = IVFAIL + 1
       IVCORR = 9
       WRITE (X02,80004) IVTNUM, IVCOMP ,IVCORR
 2281  CONTINUE
       IVTNUM = 228
C
C       ****  TEST 228  ****
C             INTEGER VARIABLE = 2 + (INTEGER VARIABLE + 4)
C
       IF (ICZERO) 32280, 2280, 32280
 2280  CONTINUE
       IVON01 = 3
       IVCOMP = 2 +(IVON01+4)
       GO TO 42280
32280  IVDELE = IVDELE + 1
       WRITE (X02,80003) IVTNUM
       IF (ICZERO) 42280, 2291, 42280
42280  IF (IVCOMP - 9) 22280, 12280, 22280
12280  IVPASS = IVPASS + 1
       WRITE (X02,80001) IVTNUM
       GO TO 2291
22280  IVFAIL = IVFAIL + 1
       IVCORR = 9
       WRITE (X02,80004) IVTNUM, IVCOMP ,IVCORR

    . . .
```

FIGURE 2

Example of Test Deletion Procedure

```
    FORTRAN COMPILER VALIDATION SYSTEM

    DEPARTMENT OF THE NAVY
    ADPE SELECTION OFFICE
    SOFTWARE DEVELOPMENT DIVISION

    PRE-RELEASE FORTRAN 1966 - LIMITED DISTRI.

    FOR OFFICIAL USE ONLY - COPYRIGHT 1975

TEST      PASS/FAIL      COMPUTED          CORRECT
--------------------------------------------------

 21          PASS
 22          PASS
 23          FAIL               0                1
 24          PASS
 25          PASS
 26          PASS
 27          PASS
 28          PASS
 29          PASS
 30          FAIL              -2                2
 31          PASS
 32          PASS

--------------------------------------------------

               END OF PROGRAM FM004

                2 ERRORS ENCOUNTERED
               10 TESTS PASSED
                0 TESTS DELETED
```

FIGURE 3

Example of Audit Routine Output Report

REFERENCES

1. Sammet, J. E., Programming Languages: History and Fundamentals, Prentice-Hall, Incorporated, 1969.

2. American Standard FORTRAN, X3.9-1966, American National Standards Institute Incorporated, New York, 1966.

3. American National Standards Committee X3J3, Draft Proposed ANS FORTRAN, ACM Sigplan Notices, Vol. 11, No. 3, 1976 March.

4. FCVS Detailed Test Specifications, available from the National Technical Information Service, US Department of Commerce, 5285 Port Royal Road, Springfield, Virginia, 22151, reference ADA030211.

5. Hopper, Captain Grace Murray, USNR, "U. S. Navy FORTRAN Tests", March, 1971, unpublished Department of the Navy Documentation.

6. Holberton, F. E. and Parker, E. G., "NBS FORTRAN Test Programs", U. S. Department of Commerce, National Bureau of Standards, NBSIP 73-250, June 1973.

7. CODASYL FORTRAN Data Base Facility, CODASYL Journal of Development, Version 1.0, August 1, 1976, published by CODASYL FORTRAN Data Base Committee.